Lesson 11

Objectives

- CPU Scheduling
- Preemptive and Non preemptive Scheduling
- Scheduling Criteria
- Scheduling Algorithms
    - FCFS
    - SJF

CPU SCHEDULING

It is always desirous to keep CPU busy in performing the task all the time. This can be carried out by a better switching technique of CPU among different processes; this ensures the efficient means of *multiprogramming*. So in order to get these goals we are in need of an efficient scheduler known as CPU scheduler. No doubt every resource of computer is scheduled before use, since sharing of resources is a key issue of operating system.

CPU-I/O Burst Cycle

Process execution consists of a *cycle* of CPU execution and I/O wait. When the CPU is executing the process called *CPU burst* when I/O is being serviced called *I/O burst*. So a process always roasted between these two bursts and ultimately terminated. Following diagram can easily show this.

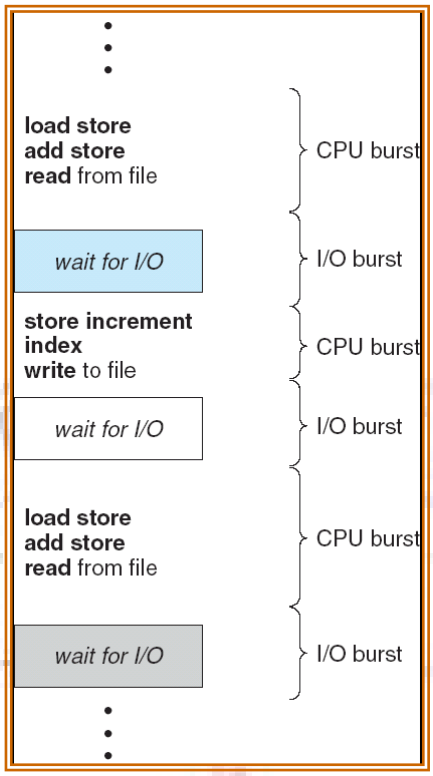
Burst Distribution

If we show the CPU activity over a histogram we can observe that it consists of:

1- Large number of *short* CPU bursts
2- Small number of *long* CPU bursts

A CPU bound process is one, which has few very long CPU bursts. This distribution is helpful in selection of appropriate CPU scheduling algorithm.

This is shown in following diagram.



CPU Scheduler

As we already have been discussing that the processes are brought into *ready queue* in main memory to allocate the processor. This queue may not be necessarily a FIFO; it could be a priority queue, a tree or a simple linked list. The records in the queue are PCBs. Short-term scheduler or CPU scheduler carries out this task. Its duties are to:

- Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them

Preemptive & non-preemptive Scheduling

- CPU scheduling decisions may take place when a process:
  1. switches from running to waiting state (for I/O request or termination of child)
  2. Switches from running to ready state (interrupt)
  3. switches from waiting to ready (after completion of I/O)
  4. Terminates

Non-preemptive

- When a new process must be selected for execution (one which is bounded to execute)
- Here is no other choice in terms of scheduling

- If the CPU is allocated to a process under this scheduling then it will release it when it will be terminated or go to waiting state
- No ready queue cause the running process to leave the CPU
- It is used by Microsoft windows
- It does not require additional hardware like timer etc

Preemptive

- CPU will be taken by the running process, if incoming is prior or existing process gets time out
- Similarly, a high priority process can preempt the CPU
- Additional timer require if CPU is assigned in time division fashion

Dispatcher

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this involves:
    - switching context
    - switching to user mode
    - jumping to the proper location in the user program to restart that program
- *Dispatch latency* – time it takes for the dispatcher to stop one process and start another running

Scheduling Criteria

- CPU utilization – keep the CPU as busy as possible
- Throughput – Number of processes that complete their execution per time unit
- Turnaround time – amount of time to execute a particular process, including execution and waiting time
- Waiting time – amount of time a process has been waiting in the ready queue
- Response time – amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment)

Optimization Criteria

- Max CPU utilization
- Max throughput
- Min turnaround time

- Min waiting time
- Min response time

SCHEDULING ALGORITHMS

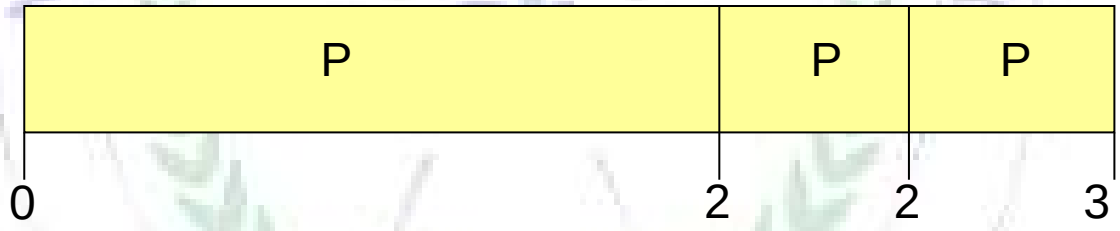
First Come First Serve (FCFS)

- It is a non-preemptive scheduling scheme
- Once the job enters in the CPU then leave it after termination or I/O
- The first job enters in ready queue will be assigned the CPU
- It has unwanted average waiting time

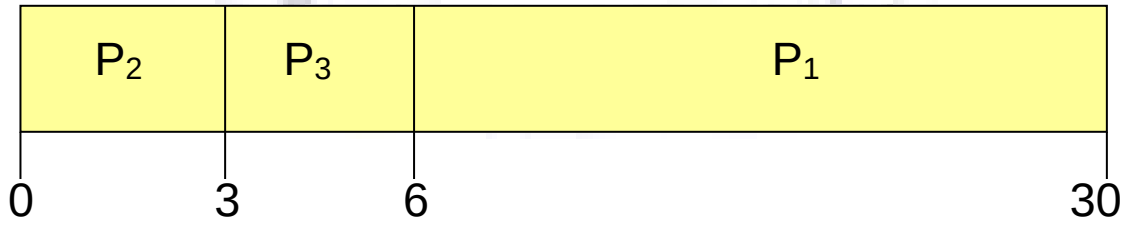
Example:

Process	Burst Time
P1	24
P2	3
P3	3

- Suppose that the processes arrive in the order: *P1, P2, P3*
  The Gantt Chart for the schedule is:



- Waiting time for *P1* = 0; *P2* = 24; *P3* = 27
- Average waiting time: (0 + 24 + 27)/3 = 17
- Suppose that the processes arrive in the order: *P2* , *P3* , *P1*



- Waiting time for *P1* = 6*; P2* = 0*; P3* = 3
- Average waiting time: (6 + 0 + 3)/3 = 3
- Much better than previous case

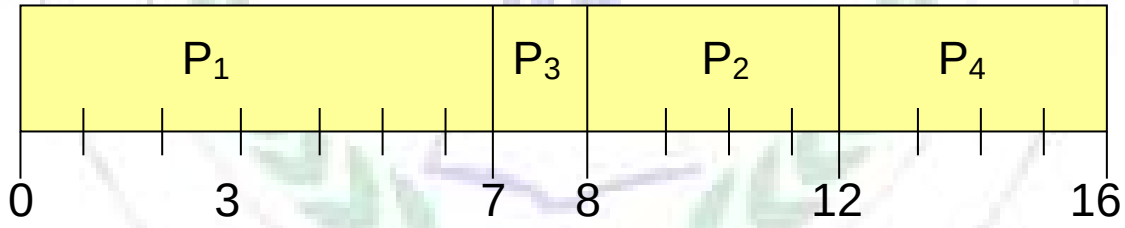
- *Convoy effect* short process behind long process

Shortest Job First (SJF) Scheduling

- Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time
- Two schemes:
  - Non-preemptive – once CPU given to the process it cannot be preempted until completes its CPU burst
  - Preemptive – if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is known as the Shortest-Remaining-Time-First (SRTF).
- SJF is optimal – gives minimum average waiting time for a given set of processes

*Example of **non-preemptive:***

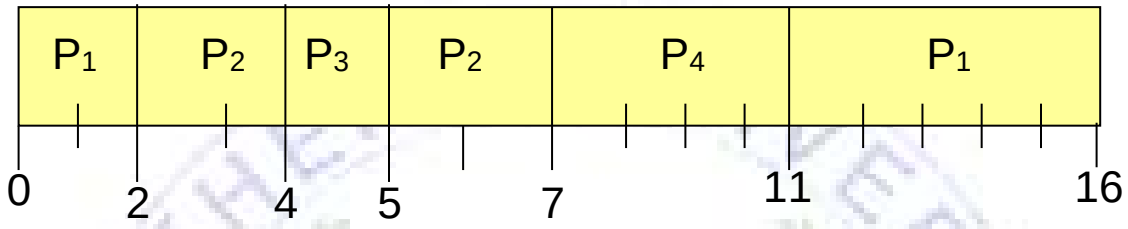
Process	Arrival Time	Burst Time
P1	0.0	7
P2	2.0	4
P3	4.0	1
P4	5.0	4



- Average waiting time = (0 + 6 + 3 + 7)/4 = 4

*Example of **preemptive**:*

Process	Arrival Time	Burst Time
P1	0.0	7
P2	2.0	4
P3	4.0	1
P4	5.0	4



- Average waiting time = (9 + 1 + 0 +2)/4 = 3
- But it is really hard to get exact next CPU burst, but we can estimate it somehow
- Can be done by using the length of previous CPU bursts, using exponential averaging

1. t_n = actual length of n^{th} CPU burst
2. τ_{n+1} = predicted value for the next CPU burst
3. $\alpha, 0 \le \alpha \le 1$ $\tau_{n=1} = \alpha\, t_n + (1-\alpha)\tau_n.$
4. Define :